Freie Universität Berlin

Volker Roth

Rechnersicherheit, SoSe 21

Übung 07

TutorIn: Oliver Wiese
Tutorium 02
Materialien: Latex, VSC, Skript

10. Juni 2021

---

1 Rainbow tables

Consider the hash function $H : \{0,\dots,9\}^{2n} \to \{0,\dots,9\}^2$ with

$$H(x_1x_2\dots x_{2n}) = y_1y_2,$$

where

$$y_1 = \left(\sum_{i=1}^{n} x_{2i-1}\right) \bmod 10 \qquad y_2 = \left(\sum_{i=1}^{n} x_{2_i}\right) \bmod 10$$

(H sums all digits on even (uneven) position modulo 10, e.g. $H(4671) = 17$.). You are given the following rainbow table with chain length 8.

start point	end point
2345	37
7033	97
4234	79
3400	11
1234	59
7455	71

The regeneration function used in every round [1] is $R : \{0,\dots,9\}^2 \to \{0,\dots,9\}^4$ with

$$R(x_1x_2) = y_1y_2y_3y_4,$$

where $y_1 = x_1, y_2 = x_2$ and

$$y_3 = (x_1 + 3) \bmod 10 \qquad y_4 = (y_2 + 5) \bmod 10.$$

For example: $R(68) = 6893$.

(a) *Find one inverse of the hash value 91 using the table above and document your steps.*

- We have two different approaches.
  1. **Solution**: Finding the inverse of the hash value $h_1 = 91$, by using R and H on $\underline{h_1}$.
     1.1 Search for $h_1 = 91$ in the table (end point column) → not found.

1.2 Use the regeneration function R on $h_1 \rightarrow$ R(91) = 9126

1.3 Use the hash function H on the result $\rightarrow$ H(9126) = 17

1.4 (And now repeat these steps until found):
Search for $h_1 = 17$ in the table (end point column) $\rightarrow$ not found.

1.5 Use the regeneration function R on $h_1 \rightarrow$ R(17) = 1742

1.6 Use the hash function H on the result $\rightarrow$ H(1742) = 59

1.7 Search for $h_1 = 59$ in the table (end point column) $\rightarrow$ found entry!

2. **Solution**: Finding the inverse of the hash value 91, by using R and H on every end point entry of the hash table.

2.1 Search for 91 in the table (end point column) $\rightarrow$ not found.

2.2 Use the regeneration function R on every entry of that column:

start point	R(end point)	end point
2345	3672	37
7033	9722	97
4234	7904	79
3400	1146	11
1234	5984	59
7455	7105	71

2.3 Use the hash function H on every result:

start point	H(R)	R	end point
2345	99	3672	37
7033	19	9722	97
4234	73	7904	79
3400	57	1146	11
1234	33	5984	59
7455	76	7105	71

2.4 (And now repeat these steps until found):
Search for 91 in the table (newly calculated hash column) $\rightarrow$ not found.

2.5 Use the regeneration function R on every entry of that column:

start point	R(H)	H	R	end point
2345	9924	99	3672	37
7033	1944	19	9722	97
4234	7308	73	7904	79
3400	5782	57	1146	11
1234	3368	33	5984	59
7455	7610	76	7105	71

2.6 Use the hash function H on every result:

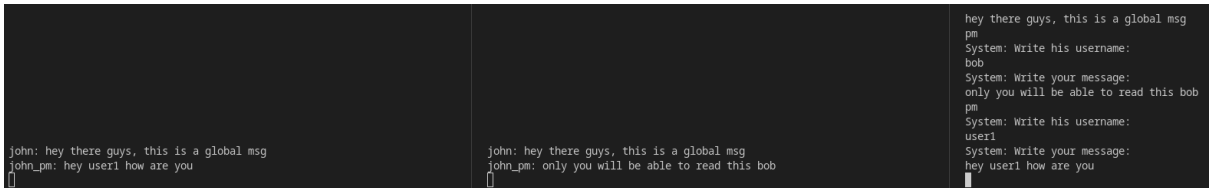
start point	H(R)	R	H	R	end point
2345	13	9924	99	3672	37
7033	53	1944	19	9722	97
4234	71	7308	73	7904	79
3400	39	5782	57	1146	11
1234	91	3368	33	5984	59
7455	77	7610	76	7105	71

2.7 Search for 91 in the table (newly calculated hash column) $\rightarrow$ entry found!

2 Your project

(a) *Personal message: A user should be able to send a personal message to another user. We still have one global chat room.*

- For this we just added a 'Command Handler' on the Server side. Example of it working:

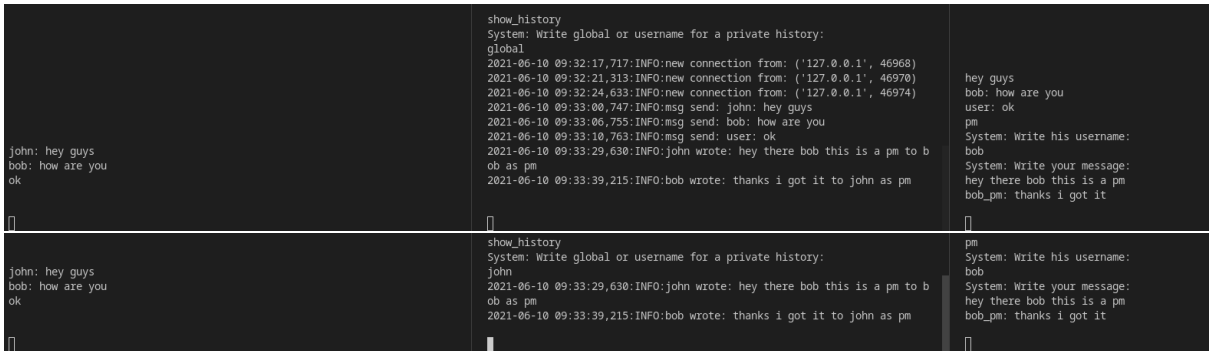


- Implementation:

```
18 def send_pm(sender: Client, connected_clients):
19     sender.socket.sendall(b'System: Write his username:')
20     username = sender.socket.recv(DATASIZE).decode('utf-8')
21     sender.socket.sendall(b'System: Write your message:')
22     pm = sender.socket.recv(DATASIZE).decode('utf-8')

24     #search for the receiver
25     for receiver in connected_clients:
26         receiver: Client
27         #if the user is online, send him the msg
28         if receiver.username == username:
29             #send the msg
30             receiver.socket.sendall(myMsg(pm, sender).format_pm())
31             #log
32             log_msg = f'{sender.username} wrote: {pm} to {receiver.username} as pm'
33             log.info(log_msg)
34             log_handler.pm_user_log(log_msg, sender.username, receiver.username)
35             break
```

(b) *History: A user should have access to a communication history (for global chat and personal messages). The history has to be stored on the server side.*

- For this we added a 'log Handler' (history), which creates/write/returns log files of users and the global log file. We also added a new function in the 'Command Handler' on the server side. Example of it working:



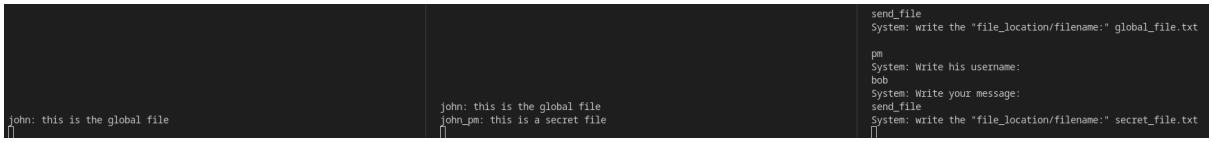
- Implementation:

```
def send_history(client: Client):
    client.socket.sendall(b'System: Write global or username for a private history:
    ')
    global_or_username = client.socket.recv(DATASIZE).decode('utf-8')

    if global_or_username == 'global':
        data = log_handler.log_to_file('log', None)
        client.socket.sendall(data)
    else:
        data = log_handler.log_to_file(client.username, global_or_username)
        client.socket.sendall(data)
```

```
import logging
log = ''

loggers = {}
def new_log(name, level = logging.INFO):
    #if logger already exists
    global loggers
    if loggers.get(name):
        return loggers.get(name)

    log = logging.getLogger(name)
    formatter = logging.Formatter('%(asctime)s:%(levelname)s:%(message)s')
    fileHandler = logging.FileHandler(f'log/{name}.log')
    fileHandler.setFormatter(formatter)
    streamHandler = logging.StreamHandler()
    streamHandler.setFormatter(formatter)

    log.setLevel(level)
    log.addHandler(fileHandler)
    log.addHandler(streamHandler)
    loggers[name] = log
    return log

log = new_log('log', logging.INFO)

def pm_user_log(log_msg, username_1, username_2):
    #sort the usernames and then join them to one filename
    filename = ''.join( sorted( (username_1, username_2) ) )
    log = new_log(filename)
    log.info(log_msg)

def log_to_file(log_name_1, log_name_2):
    if log_name_2 == None:
        with open('log/log.log', 'rb') as file:
            return file.read()

    else:
        filename = ''.join( sorted( (log_name_1, log_name_2) ) )
        filename = f'log/{filename}.log'
        try:
            with open(filename, 'rb') as file:
                return file.read()
        except:
            return b'No such history'
```

(c) *Attachments: A user should be able to send files to other users and the chat room. Files should be stored on the server side (as part of the history) and client side (such that the user can open the file).*

- As for now the file is not saved on the client side. It will only print on his console. For this we added a 'Command Handler' on the client side. Example of it working:



- Implementation:

```
15 def send_file(server_socket):
16     location = input('System: write the "file_location/filename:" ')
17     #open file and send it
18     with open(location, 'rb') as file:
19         server_socket.send(file.read())
```